INFORMATION AND COMPUTATION **129**, 53–62 (1996)
ARTICLE NO. 0073

Kalvis Apsītis

Department of Computer Science, University of Maryland, College Park, Maryland 20742
E-mail: kalvis@cs.umd.edu

Rūsiņš Freivalds

Institute of Mathematics and Computer Science, University of Latvia, Raiņa bulvāris 29, Rīga LV-1459, Latvia
E-mail: rusins@cclu.lv

and

Carl H. Smith

Department of Computer Science, University of Maryland, College Park, Maryland 20742
E-mail: smith@cs.umd.edu

Previous work in inductive inference dealt mostly with finding one or several machines (IIMs) that successfully learn collections of functions. Herein we start with a class of functions and consider *the learner set* of all IIMs that are successful at learning the given class. Applying this perspective to the case of team inference leads to the notion of *diversification* for a class of functions. This enable us to distinguish between several flavours of IIMs all of which must be represented in a team learning the given class.

1. INTRODUCTION

All current theoretical approaches to machine learning tend to focus on a particular machine or a collection of machines and then find the class of concepts which can be learned by these machines under certain constraints defining a criterion of successful learning [AS83, OSW86]. In this paper we investigate the dual problem: Given some set of concepts, which algorithms can learn all those concepts?

From [AGS89] we know that in the theory of inductive inference sometimes one concept must be mastered before the learning of another concept can be initiated. This observation is consistent with common human learning behavior. For example, learning how to walk is something that most people master very early in their life. A smaller number of us learn how to drive a car, while even fewer of us learn how to pilot an airplane. Although there may be counterexamples, it seems safe to assert that those who can pilot an aircraft have also learned how to walk or how to operate an automobile. Based on this example, one would expect that some concepts are learned by more machines than others. Our results indicate that this is indeed the case, but only if instead of a single concept we consider a suitable infinite set of concepts.

Team learning has been a prevalent theme in the study of inductive inference [Smi94b]. An early result asserts that the larger the team allowed, the larger the class of learnable sets of functions becomes [Smi82]. This suggests that perhaps different types of knowledge are needed to solve some problems. Indeed, most of the papers in our field have at least two coauthors. A precise correspondence between team learning and probabilistic learning [Pit89] intensified the study of team learning. So far, all of the studies of team learning focus on how team size compares with other parameters relevant to the learning process [Sch86, PS88, FSV89, KZ91, DPVW91, DKV92, DKV93]. The goal has always been to see which parameter settings allowed more powerful teams of learning machines to be constructed.

In contrast, we address the issue of how to compose teams. Starting with the common observation that humans display a broad range of learning proclivities, we hypothesize that by considering the collection of all learning algorithms we should be able to distinguish between "flavors" of learning algorithms. While we still cannot name these flavors, nor describe them intuitively, it is possible to determine how many of them are necessary to perform some learning tasks. For example, we start with an arbitrary set of functions that is not learnable by a single machine. Then we show how to partition the set of all learning algorithms into two families such that it is impossible to find a finite team of any size learning the original set if we only choose learning algorithms from one of the families. The partitions

* Results collected in this paper were presented previously at conferences [AFS94] and [FS93].

0890-5401/96 $18.00

of machines into families, as described above, are called diversifications.

Section 3 provides the background of team learning. Sections 4 and 5 show the dual approach applied to single functions and those classes of functions which are learnable by a single machine. Section 6 concentrates on classes unlearnable by a single IIM.

2. PRELIMINARIES

The set of all natural numbers is denoted by $\mathbb{N}$, the set of all single argument recursive functions by $\mathcal{F}$, and the set of partial recursive functions by $\mathcal{P}$ [Rog67]. Letters $h, i, j, k, l, m, n, x, y$ vary over $\mathbb{N}$, f and g over $\mathcal{F}$, and φ, ψ over $\mathcal{P}$. Classes of recursive functions, i.e., subsets of $\mathcal{F}$, are denoted by U, V, W, with or without decorations. By using standard encoding techniques, the natural numbers can serve as indices for programs [MY78, Smi94a]. The function computed by program i is denoted by φ_i. The collection of functions $\varphi_0, \varphi_1, \varphi_2, \ldots$, forms an *acceptable programming system.* Some fixed recursive one–one onto mapping between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$ will be denoted by $\langle \cdot, \cdot \rangle$. This encoding can also be extended to a variable number of arguments: $\langle x_1, \ldots, x_n \rangle$, where $x_i, n \in \mathbb{N}$.

We use the notation of first order logic. The quantifier $\overset{\infty}{\forall}$ is read "for all but finitely many" and $\exists!$ is read "there is exactly one." We also use set theoretical notation. Relations $\in$, $\subseteq$, $\subset$ denote "is an element of," "is a subset of," "is a proper subset of," respectively. $\cup$ and $\cap$ denote set union and intersection. If f is a mapping from X to Y, $A \subseteq X$ and $B \subseteq Y$, then $f(A)$ and $f^{-1}(B)$ denote *image* and *inverse image*, respectively. Set of all arguments for which φ is defined is denoted $\mathrm{Dom}(\varphi)$. Notation $\varphi(x)\downarrow$ means $x \in \mathrm{Dom}(\varphi)$, and $\varphi(x)\uparrow$ means $x \notin \mathrm{Dom}(\varphi)$. Relation $\varphi \preccurlyeq \psi$ means that φ is a subfunction of ψ, i.e. $\varphi(x) = y$ always implies $\psi(x) = y$. Mappings from $\mathbb{N}$ to other sets are called sequences and denoted either by a list of members: $a_1, a_2, \ldots$, or simply $(a_n)_{n \in \mathbb{N}}$.

Inductive inference machines (called also learning machines or IIMs) are denoted by M, with or without decorations. They can be partial or total. The functions to be learned by IIMs in appropriate context are called also *targets* of learning. Arguments for learning machines are *strings* of ordered pairs $\langle x, f(x) \rangle$. During the learning process machine M receives a sequence of strings, each an extension of the previous one,

$$\sigma_1 = \langle \langle x_1, f(x_1) \rangle \rangle,$$
$$\sigma_2 = \langle \langle x_1, f(x_1) \rangle, \langle x_2, f(x_2) \rangle \rangle,$$
$$\ldots$$

and outputs sequence of *conjectures* $h_1 = M(\sigma_1)$, $h_2 = M(\sigma_2), \ldots$. If M is partial, some of the conjectures may be undefined. Each string σ_n defines some finite function, and we have $\sigma_1 \preccurlyeq \sigma_2 \preccurlyeq \cdots$. We require that the sequence $\sigma_1, \sigma_2, \ldots$ *expands* to the target f, written $(\sigma_n) \nearrow f$, meaning that for any x from the domain of f, the pair $(x, f(x))$ appears in the growing sequence $\sigma_1 \preccurlyeq \sigma_2 \preccurlyeq \cdots$ starting from some place.

Now we are going to define several multivalued mappings called *identification types* from IIMs to functions. If $\mathcal{L}$ is such a mapping, its meaning can be expressed intuitively as follows: "Machine M *identifies* (or *learns*) all functions $f \in \mathcal{L}(M)$ in the sense of type $\mathcal{L}$." In this paper $\mathcal{L}$ will denote any one of the types EX, EX_n, FIN to be defined below.

DEFINITION 1 [Gol67]. Machine M learns recursive function f *in the limit*, written $f \in EX(M)$, if for any sequence of input strings $(\sigma_n) \nearrow f$ we have sequence of conjectures $M(\sigma_n)$ converging to some correct index h of f, i.e.,

$$(\exists h)(\exists n_0)[M(\sigma_{n_0}) = h \text{ and } (\varphi_h = f) \text{ and } (\forall n > n_0)(M(\sigma_n) = h \text{ or } M(\sigma_n)\uparrow)].$$

DEFINITION 2. Machine M commits n mind changes on the sequence $(\sigma_k) \nearrow f$ if there is a subsequence $\sigma_{k_1}, \sigma_{k_2}, \ldots, \sigma_{k_{n+1}}$ such that $M(\sigma_{k_i})\downarrow$, $i = 1, \ldots, n+1$, and $M(\sigma_{k_i}) \neq M(\sigma_{k_{i+1}})$, $i = 1, \ldots, n$. Moreover, we demand that there be no subsequence of length $n+2$ with the same property.

DEFINITION 3 [CS83]. Machine M learns a recursive function f *in the limit with a mind change bound n*, written: $f \in EX_n(M)$, if $f \in EX(M)$ and the number of mind changes, committed by M on any $(\sigma_k) \nearrow f$ does not exceed n.

The type EX_0 is of special importance, we denote it also FIN and write $f \in FIN(M)$ to means "M finitely learns f" or "M is a one shot learner of f." In this case M outputs only one but correct conjecture.

Symbols of identification types are typically overloaded. Not only do they denote multivalued mappings (or binary relations between IIMs and functions), but they also stand for collections of classes of functions identifiable by a single machine. More precisely

$$EX = \{ U \subseteq \mathcal{F} \mid (\exists M)[U \subseteq EX(M)] \}.$$

Similar definitions for EX_n and FIN should be clear.

Let $\mathcal{L}$ be any identification type. For a set of IIMs $\mathcal{T}$ we denote all functions that are learned by at least one IIM in $\mathcal{T}$ by *image* $\mathcal{L}(\mathcal{T})$ defined as

$$\mathcal{L}(\mathcal{T}) = \bigcup_{M \in \mathcal{T}} \mathcal{L}(M).$$

Let $\mathscr{L}^{-1}(f)$ denote the set of all machines identifying function f. Then for a set U of recursive functions we define

$$\mathscr{L}^{-1}(U)=\bigcup_{f\in U}\mathscr{L}^{-1}(f),$$

the *inverse image*, i.e., the set of all machines learning at least one function in U. This notation is reasonable if we consider $\mathscr{L}$ as multivalued mapping from IIMs to functions. In Section 5 we introduce another pair of dual notions, namely learner sets and foci.

In some places we will consider $\mathscr{P}$ and $\mathscr{F}$ as topological spaces, not just sets. The rest of definitions in this section refer to partial as well as to total functions. Let $\mathscr{X}$ be the notation for either $\mathscr{P}$ or $\mathscr{F}$.

DEFINITION 4. For a function $\varphi\in\mathscr{X}$ and a finite function $\sigma\preccurlyeq f$ the neighborhood of φ determined by σ is $U_\varphi^\sigma=\{\psi\in\mathscr{X}\mid\sigma\preccurlyeq\psi\}$.

We can think of some neighborhood of φ as set of all functions "near" to φ in the sense, that they cannot be distinguished from φ by receiving some finite input, for example, input σ. We may omit the superscript in U_φ^σ.

DEFINITION 5. Set $A\subseteq\mathscr{X}$ is *closed* if any $\psi\notin A$ has a neighborhood U_ψ disjoint from A.

As an immediate consequence of Definition 5, the relation "$\psi\notin A$" can be established after seeing finitely many values of ψ whenever A is a closed set of functions not containing ψ.

3. TEAM INFERENCE

DEFINITION 6. Class U of recursive functions is identifiable by k machines from a team $M_1, ..., M_n$, written $U\in[k,n]\,\mathscr{L}$, if for any $f\in U$, $f\in\mathscr{L}(M_j)$ for at least k different $j\in\{1, ..., n\}$.

Note that for different functions from U, different collections of machines from the team may succeed. $[k,n]\mathscr{L}$ itself is an identification type along with $\mathscr{L}$ for any choice of $k\leqslant n$. In the two extreme cases we have $[n,n]\mathscr{L}=\mathscr{L}$ and $[0,n]\,\mathscr{L}=2^{\mathscr{F}}$, i.e. the collection of all classes of recursive functions. All $[k,n]\,\mathscr{L}$ are called types *derived* from $\mathscr{L}$.

For a fixed $\mathscr{L}$ some of the derived types $[k,n]\,\mathscr{L}$ are essentially different. For instance, $EX\subset[1,2]\,EX$ (cf. Example 10). It turns out that all $[1,m]\,EX$ are different and all the other types derived from EX by team inference are reducible to them. The complete picture for the case EX is given by the following two theorems:

THEOREM 7 [Smi82]. *For any* $m>0$, $[1,m]\,EX\subset[1,m+1]\,EX$.

THEOREM 8 [PS88]. *For any* $n\geqslant m>0$, $[m,n]\,EX=[1,\lfloor n/m\rfloor]\,EX$.

For the types EX_n the picture of team inclusions is more complicated. We shall need these two properties:

THEOREM 9 [KF92, AFK$^+$92]. *For any* $n\geqslant 0$, $EX_n\subset[2^{n+2}-2, 2^{n+2}-1]\,EX_n$ *and* $EX_n=[2^{n+2}-1, 2^{n+2}]\,EX_n$.

In particular, $FIN\subset[2,3]\,FIN$ and hence $FIN\subset[1,2]\,FIN$, but $FIN=[3,4]\,FIN$. For relations $EX\subset[1,2]\,EX$ and $FIN\subset[1,2]\,FIN$ we give classic illustrations:

THEOREM 10 [Bar74, BB75]. *There exist classes* $U, V\subset\mathscr{F}$ *such that* $U, V\in EX$ *but* $U\cup V\notin EX$.

Let $U=\{f\in\mathscr{F}\mid\overset{\infty}{\forall}x(f(x)=0)\}$ to be the functions of finite support and let $V=\{f\in\mathscr{F}\mid\varphi_{f(0)}=f\}$ be the self-describing functions. It is easy to verify that they both are in EX, but their union is not. Therefore, $U\cup V\in[1,2]\,EX$ and $U\cup V\notin EX$.

THEOREM 11 [Wie74]. *There are a class* $U\subset\mathscr{F}$ *and a function* f *such that* $U\in FIN$ *and* $U\cup\{f\}\notin FIN$.

One such example is given below:

$$f_n(x)=\begin{cases}1 & \text{if } n=x,\\ 0 & \text{otherwise;}\end{cases}\qquad U=\{f_n\mid n\in\mathbb{N}\};\quad f(x)=\lambda x[0].$$

Clearly, $U\cup\{f\}\in[1,2]\,FIN$.

Therefore, adding one function to a class can change its FIN-learnability. Type EX is more robust. Here are two results showing those modifications which do not change EX-learnability of a class.

THEOREM 12. *If* $U=EX(M)$ *and* V *is a.r.e. closed class of recursive functions, then there exists a machine* M' *such that* $U\cup V=EX(M')$.

Proof. Let v be a generator for V such that $V=\{\varphi_{v(0)}, \varphi_{v(1)}, ...\}$. Our machine maintains a counter i for the set V which at any moment is set to the minimal value such that $\varphi_{v(i)}$ does not contradict the input seen so far.

ALGORITHM OF M'.
receive input σ;
if $\sigma\preccurlyeq\varphi_{v(i)}$ then {output $v(i)$;} else {increment i; output $M(\sigma)$;}

Since all $\varphi_{v(i)}$ are total, tests $\sigma\preccurlyeq\varphi_{v(i)}$ can be done effectively. Suppose $f\in U\cup V$. If $f\in V$, then the counter stops at the right place in the list of V, and M' converges to a correct conjecture $v(i)$. If $f\notin V$, then, since V is closed, there exists a neighborhood of f disjoint from V. Starting from some moment all the inputs fall in this neighborhood, i.e., $\sigma\not\preccurlyeq\varphi_{v(i)}$. Machine M' then increments i on each step and behaves like M. In both cases f is identified. Clearly, M' does not identify any functions outside $U\cup V$. ∎

Recall the classes U and V given in the comment after Theorem 10. Since U is r.e. but not closed, and V is closed but not r.e., we cannot drop either condition for class V in Theorem 12 (see Conclusions). A widely known result follows from Theorem 12 as a particular case: EX-learnability of a class does not change if we add to it any finite number of recursive functions.

THEOREM 13. *If $U = EX(M)$ and V is a r.e. closed class, then there exists a machine M' such that $U - V = EX(M')$.*

Proof. Modify the proof of Theorem 12, so that M' outputs an index of empty function rather than indices of functions in V whenever $\sigma \preccurlyeq \varphi_{v(i)}$. ∎

4. AN ISOMORPHISM RESULT

Machines may differ considerably as to the size of the class of functions they can learn. Consider the dual task: For a given function f, find the class of machines which learn it. We find that all such families $EX^{-1}(f)$ are pairwise isomorphic accordingly to Definition 20. The definition of *recursive operator* can be found in [Rog67].

THEOREM 14 [Rog67]. *Let $\theta: \mathscr{P} \to \mathscr{P}$ be a recursive operator and f and g partial recursive functions. Then θ has the following properties*:

1. *Monotonicity.* $g \preccurlyeq h \Rightarrow \theta(g) \preccurlyeq \theta(h)$.
2. *Continuity.* $(x, y) \in \theta(f) \Rightarrow (\exists \sigma$: *finite function*) $(\sigma \preccurlyeq f$ *and* $(x, y) \in \theta(\sigma))$.
3. *Existence of a witness. There is a recursive function $\bar{\theta}$, a witness, such that for all x: $\varphi_{\bar{\theta}(x)} = \theta(\varphi_x)$. (Clearly, $\bar{\theta}$ has the* extensional property: $\varphi_x = \varphi_x \Rightarrow \varphi_{\bar{\theta}(x)} = \varphi_{\bar{\theta}(y)}$.)

If a recursive operator θ happens to be a one–one and onto mapping and its inverse is also a recursive operator, we denote it by θ^{-1}. The following two propositions easily follow from the monotonicity and continuity of θ and θ^{-1}.

COROLLARY 15. *Let θ be some recursive operator which has an inverse θ^{-1}. Then for every finite function f, the image $\theta(f)$ is also a finite function whose domain has the same cardinality as domain of f.*

COROLLARY 16. *Let θ be a recursive operator. Then for any sequence of finite functions $(\sigma_n)_{n \in \mathbb{N}} \nearrow f$ we have also $(\theta(\sigma_n))_{n \in \mathbb{N}} \nearrow \theta(f)$.*

LEMMA 17. *Let θ be a recursive operator for which an inverse θ^{-1} exists. Then the respective witnesses $\bar{\theta}$ and $\overline{\theta^{-1}}$ can be chosen to be recursive permutations, i.e., recursive one to one mappings of $\mathbb{N}$ onto itself.*

Proof. Let g, h be arbitrary witnesses for θ and θ^{-1} respectively. They can be easily made one to one by padding [Rog67]. Assume that θ maps $\alpha \in \mathscr{P}$ to $\beta \in \mathscr{P}$. Let A, B be the collections of indices for the functions α and β respectively. We have

$$g(A) \subseteq B, h(B) \subseteq A.$$

Accordingly to the Myhil isomorphism theorem we can construct a recursive permutation $\bar{\theta}$ such that $\bar{\theta}(A) = B$. Moreover, $\bar{\theta}$ in this construction depends only upon g and h, and not upon the particular sets A, B. Therefore $\bar{\theta}$ is a witness for the operator θ. Its inverse $\bar{\theta}^{-1}$ is also a recursive permutation, and we can choose $\overline{\theta^{-1}} = \bar{\theta}^{-1}$. ∎

From now on the witnesses $\bar{\theta}$ and $\overline{\theta^{-1}}$ will be chosen in accordance with this lemma. Any learning machine M receives in the input a finite function σ and returns a natural number. The composition $\overline{\theta^{-1}} M \theta$ does the same thing in accordance with Corollary 15 and therefore can also be considered as a learning machine.

COROLLARY 18. *Let θ be a recursive operator for which θ^{-1} exists and let Φ_θ be an operator on the collection $\mathscr{T}$ of all learning machines defined by*

$$\Phi_\theta(M) = \overline{\theta^{-1}} M \theta, \quad M \in \mathscr{T}. \tag{1}$$

Then Φ_θ is a one to one mapping of $\mathscr{T}$ onto itself.

Corollaries 16 and 18 imply the following theorem.

THEOREM 19. *Let $M_2 = \Phi_\theta(M_1)$, where Φ_θ is such as defined by* (1). *Then for any $f \in \mathscr{F}$ we have $f \in EX(M_2)$ iff $\theta(f) \in EX(M_1)$.*

DEFINITION 20. Two families of machines $\mathscr{T}_1$, $\mathscr{T}_2$ are *isomorphic* if there exists a recursive operator θ such that Φ_θ is a bijection between $\mathscr{T}_1$ and $\mathscr{T}_2$.

THEOREM 21. *$EX^{-1}(f_1)$ and $EX^{-1}(f_2)$ are isomorphic for any $f_1, f_2 \in \mathscr{F}$.*

Proof. We can take

$$\theta(g)(x) = \begin{cases} f_2(x) & \text{if } g(x) = f_1(x) \\ f_1(x) & \text{if } g(x) = f_2(x) \\ g(x) & \text{otherwise.} \end{cases}$$

Clearly, $\theta(f_1) = f_2$ and $\theta(f_2) = f_1$. Moreover, θ is inverse to itself, i.e., $\theta^{-1} = \theta$. Use Theorem 19 to verify that $\Phi_\theta(EX^{-1}(f_1)) = EX^{-1}(f_2)$. ∎

5. SETS OF LEARNERS AND FOCI

For each identification type $\mathscr{L}$ we introduce two mutually dual notions. Given a set U of recursive functions, the *learner set* of U, written $L_{\mathscr{L}}(U)$, is the collection of all IIMs that can $\mathscr{L}$-learn all the functions in U:

$$L_{\mathscr{L}}(U)=\{M \mid U\subseteq \mathscr{L}(M)\}=\bigcap_{f\in U}\mathscr{L}^{-1}(f).$$

For a set of IIMs $\mathscr{T}$, the *focus* of $\mathscr{T}$, written $F_{\mathscr{L}}(\mathscr{T})$, is the set of all recursive functions that are learned by all the IIMs in $\mathscr{T}$:

$$F_{\mathscr{L}}(\mathscr{T})=\bigcap_{M\in\mathscr{T}}\mathscr{L}(M).$$

By definition, the focus of the empty set of IIMs is the whole set of recursive functions, and the learner set of the empty class—the collection of all partial recursive strategies. We begin with simple set theoretical properties:

LEMMA 22 (Monotonicity). *$\mathscr{L}$ denotes some identification type. Let $\mathscr{T}_1\subseteq\mathscr{T}_2$ be two collections of machines and $U_1\subseteq U_2$ two classes of functions. Then*

1. $F_{\mathscr{L}}(\mathscr{T}_1)\supseteq F_{\mathscr{L}}(\mathscr{T}_2)$,
2. $L_{\mathscr{L}}(U_1)\supseteq L_{\mathscr{L}}(U_2)$.

LEMMA 23. *Let $\mathscr{L}$ be some identification type. By $\{\mathscr{T}_i\}_{i\in\Lambda}$ and $\{U_i\}_{i\in\Lambda}$ we denote collections of sets of machines and (partial) recursive functions respectively with some finite or infinite set of indices Λ.*

1. $F_{\mathscr{L}}(\bigcup_{i\in\Lambda}\mathscr{T}_i)=\bigcap_{i\in\Lambda}F_{\mathscr{L}}(\mathscr{T}_i)$,
2. $F_{\mathscr{L}}(\bigcap_{i\in\Lambda}\mathscr{T}_i)\supseteq\bigcup_{i\in\Lambda}F_{\mathscr{L}}(\mathscr{T}_i)$,
3. $L_{\mathscr{L}}(\bigcup_{i\in\Lambda}U_i)=\bigcap_{i\in\Lambda}L_{\mathscr{L}}(U_i)$,
4. $L_{\mathscr{L}}(\bigcap_{i\in\Lambda}U_i)\supseteq\bigcup_{i\in\Lambda}L_{\mathscr{L}}(U_i)$,
5. $F_{\mathscr{L}}(L_{\mathscr{L}}(U))\supseteq U$,
6. $L_{\mathscr{L}}(F_{\mathscr{L}}(\mathscr{T}))\supseteq\mathscr{T}$.

Proof. Use the monotonicity of $F_{\mathscr{L}}$ and $L_{\mathscr{L}}$ respectively. ∎

For identification types such as EX, EX_n, and FIN, it is easy to come up with examples where inclusions for items 2 and 4 in the above lemma are proper. Concerning items 5 and 6 we introduce two new notions.

DEFINITION 24. Set of functions U is $\mathscr{L}$-full if $F_{\mathscr{L}}(L_{\mathscr{L}}(U))=U$.

DEFINITION 25. Collection of machines $\mathscr{T}$ is $\mathscr{L}$-maximal if $L_{\mathscr{L}}(F_{\mathscr{L}}(\mathscr{T}))=\mathscr{T}$.

We proceed with some illustrations.

THEOREM 26. *U is EX-full iff $U\in EX$ or $U=\mathscr{F}$.*

Proof. "$\Rightarrow$." Suppose that U is EX-full. $U\notin EX$ implies $L_{EX}(U)=\varnothing$ and hence $F_{EX}(L_{EX}(U))=\mathscr{F}$.

"$\Leftarrow$." $L_{EX}(\mathscr{F})=\varnothing$, so $F_{EX}(L_{EX}(\mathscr{F}))=\mathscr{F}$ and $\mathscr{F}$ is full. Assume that $U\in EX$ and M is some machine learning U. If $U=EX(M)$, then $M\in L_{EX}(U)$; using monotonicity yields $U=F_{EX}(\{M\})\supseteq F_{EX}(L_{EX}(U))$ and we are done. If $U\subset EX(M)$, then in accordance with Theorem 13 for any function $f\in EX(M)-U$ we can construct a machine M_f such that $EX(M_f)=EX(M)-\{f\}$. The set of all these machines is a subset of $L_{EX}(U)$ and its focus is exactly U. We conclude that $F_{EX}(L_{EX}(U))\subseteq U$, which together with item 5 of Lemma 23 completes the proof. ∎

COROLLARY 27. $\forall U\subseteq\mathscr{F}$, $L_{EX}(F_{EX}(L_{EX}(U)))=L_{EX}(U)$.

Proof. $U\notin EX$ implies that both $L_{EX}(F_{EX}(L_{EX}(U)))$ and $L_{EX}(U)$ are empty. If $U\in EX$, we have that U is EX-full which implies the corollary. ∎

COROLLARY 28. *For any collection of machines $\mathscr{T}$ we have $F_{EX}(L_{EX}(F_{EX}(\mathscr{T})))=F_{EX}(\mathscr{T})$.*

Proof. $\mathscr{T}=\varnothing$ implies $F_{EX}(L_{EX}(F_{EX}(\mathscr{T})))=F_{EX}(\mathscr{T})=\mathscr{F}$. $\mathscr{T}\neq\varnothing$ implies that $F_{EX}(\mathscr{T})\in EX$ and we can apply Theorem 26. ∎

THEOREM 29. *A collection of machines $\mathscr{T}$ is EX-maximal iff there is U such that $\mathscr{T}=L_{EX}(U)$.*

Proof. "$\Rightarrow$." Consider $U=F_{EX}(\mathscr{T})$. Since $\mathscr{T}=L_{EX}(F_{EX}(\mathscr{T}))$, we have $\mathscr{T}=L_{EX}(U)$.

"$\Leftarrow$." Corollary 27 gives $L_{EX}(F_{EX}(L_{EX}(U)))=L_{EX}(U)$, so $\mathscr{T}=L_{EX}(U)$ is EX-maximal. ∎

For any set $W\subseteq\mathscr{F}$ denote $F_{EX}(L_{EX}(W))$ by $\bar{W}$. Let U, V be arbitrary classes of recursive functions. We summarize some of the previous results:

- $U\subseteq\bar{U}$,
- $U\subseteq V\Rightarrow\bar{U}\subseteq\bar{V}$,
- $\bar{U}=\bar{\bar{U}}$.

But $\overline{U\cup V}\neq\bar{U}\cup\bar{V}$, therefore $U\to\bar{U}$ is not a closure operator [Eng89]. To see that $\overline{U\cup V}\neq\bar{U}\cup\bar{V}$, consider $U, V\in EX$ such that $U\cup V\notin EX$, as in Theorem 10. Then $\overline{U\cup V}=\mathscr{F}$, but $\bar{U}\cup\bar{V}=U\cup V\neq\mathscr{F}$. Therefore the relation $EX\subset[1,2]\,EX$ upsets this first attempt to introduce useful dual notions. Another approach will be given in the next section.

6. DIVERSIFICATIONS

Consider the class $W=U\cup\{f\}$, where U and f are defined in the comment after Theorem 11. Clearly, $W\notin FIN$ and $W\in[1,2]\,FIN$, i.e., W can be learned by a team of 2 machines M_1, M_2. The obvious choices for M_1 and M_2 result in a pair of radically different machines. There is a kind of IIM waiting indefinitely for the value "1" and failing to identify the everywhere zero function $f\in W$. Another kind of IIM identifies the zero function, but fails to identify all but finitely many functions from $U \subset W$. Any team learning W should contain machines from either kind.

At first it seems, that this phenomenon is caused by the structure of W as the function f is the only accumulation point of the class W. We shall see, however, that for any class W which can only be identified by a team starting from some size $n > 1$, all machines useful for the team identification of W in the sense EX can be split into several kinds or flavors. Collection of different flavors of machines, which are all needed to identify W, reflect the inherent and necessary diversity of learning algorithms.

6.1. *General Results*

By a *b-partition* of set $\mathscr{T}$ we mean a collection $\{\mathscr{T}_1, ..., \mathscr{T}_b\}$ of subsets of $\mathscr{T}$ such that:

1. $\mathscr{T}_i \cap \mathscr{T}_j = \varnothing$ when $i \neq j$,
2. $\bigcup \mathscr{T}_i = \mathscr{T}$.

DEFINITION 30. Let W be a class of functions. Let $\{\mathscr{T}_1, ..., \mathscr{T}_b\}$ be a b-partition of some family $\mathscr{T}$ of machines. The b-partition is called (a, b)-diversification for W if any finite team $\mathscr{T}' \subseteq \mathscr{T}$ learning W intersects at least a of the b families $\mathscr{T}_i$.

Only cases where $2 \leqslant a \leqslant b$ make sense. A finite team $\mathscr{T}' = \{M_1, ..., M_n\}$ $\mathscr{L}$-learns a class W whenever $W \subseteq \mathscr{L}(\mathscr{T}') = \mathscr{L}(M_1) \cup \cdots \cup \mathscr{L}(M_m)$; more precisely, it is $[1, m]$ $\mathscr{L}$-learning in the sense of Definition 6. It is easier to prove the existence of an (a, b)-diversification first for finite collections of machines $\mathscr{T}'$ only. The following lemma enables us to proceed to diversifications for infinite families of machines as well, provided that all their finite subfamilies have a diversification.

LEMMA 31 (König, [Kur58]). *In any infinite directed tree in which each node has only a finite number of direct successors, there is an infinite path proceeding from the root.*

LEMMA 32. *Let W be a class of functions. There exists an (a, b)-diversification of an infinite family $\mathscr{T}$ of machines (in particular of the set of all recursive machines) for the class W iff any finite subfamily $\mathscr{T}' \subset \mathscr{T}$ has an (a, b)-diversification for W.*

Proof. "$\Rightarrow$." Assume that there is no (a, b)-diversification for some $\mathscr{T}' \subset \mathscr{T}$. Then no b-partition of the larger collection $\mathscr{T}$ can be an (a, b)-diversification for W.

"$\Leftarrow$." Let $M_1, M_2, \ldots$ be a list of all machines from $\mathscr{T}$. We build and infinite *diversification tree*. Nodes in this tree are denoted by $v_{\langle i_1 \cdots i_n \rangle}$, where $n \geqslant 0$ and $i_j \in \{1, ..., b\}$. Any such $v_{\langle i_1 \cdots i_n \rangle}$ corresponds to some b-partition of the finite family of machines $\mathscr{T}' = \{M_1, ..., M_n\}$. Namely, we define a b-partition of $\mathscr{T}'$:

$$\mathscr{T}_k = \{M_j \in \mathscr{T}' \mid i_j = k\} \qquad \text{for all} \quad k = 1, ..., b.$$

The diversification tree contains only those nodes whose b-partitions are (a, b)-diversifications for the class W. The tree is rooted at the node $v_{\langle\rangle}$ which corresponds to the empty (a, b)-diversification of the empty family of machines. It has downward edges between any pair of nodes $v_{\langle i_1 \cdots i_{n-1} \rangle}$ and $v_{\langle i_1 \cdots i_{n-1} i_n \rangle}$ which have the first $n-1$ indices in common.

An example of $(2, 2)$-diversification tree is shown in Fig. 1. The node $v_{\langle 1, 2 \rangle}$ does not have the right successor, meaning that the team $\{M_2, M_3\}$ identifies W, and therefore $v_{\langle 1, 2, 2 \rangle}$ does not represent a $(2, 2)$-diversification of $\{M_1, M_2, M_3\}$. Any diversification tree is symmetric, e.g. the presence of $v_{\langle 1, 1, 2 \rangle}$ implies the presence of $v_{\langle 2, 2, 1 \rangle}$ in Fig. 1.

This tree is infinite, since for each $n \geqslant 0$ there exists some k-diversification of the family $\mathscr{T}' = \{M_1, ..., M_n\}$; i.e., there is a path of any given length n. Moreover, any vertex in the tree has no more than b successors. We can apply Lemma 31 and obtain an infinite path $v_{\langle\rangle}, v_{\langle i_1 \rangle}, v_{\langle i_1 i_2 \rangle}, \ldots$. The sequence $i_1, i_2, \ldots$ defines an (a, b)-diversification on $\mathscr{T}$. ∎

THEOREM 33. *Let $\mathscr{L}$ be some identification type, W some class of functions and $b \geqslant 2$. The following statements are equivalent*:

1. *There is no (b, b)-diversification of all recursive machines for the class W.*

2. *W can be expressed as finite union $W = W_1 \cup \cdots \cup W_n$ such that*

$$\forall S \subseteq \{1, ..., n\} \left(|S| \leqslant b \Rightarrow \bigcup_{i \in S} W_i \in [1, b-1] \, \mathscr{L} \right).$$

Proof. $(1) \Rightarrow (2)$ Assume that the collection of all IIMs $\mathscr{T}$ has no (b, b)-diversification for W. Lemma 32 implies the existence of a finite collection of machines $\mathscr{T}' = \{M_1, ..., M_m\}$ which has no (b, b)-diversification for W. Notice that the team $\mathscr{T}'$ itself learns W, for otherwise any b-partition of $\mathscr{T}'$ were (b, b)-diversification. Split the set W into a union of equivalence classes $W_1, ..., W_n; f, g, \in W$ are equivalent iff $f \in \mathscr{L}(M_j)$ implies $g \in \mathscr{L}(M_j)$ and vice versa for all $M_j \in \mathscr{T}$. We have $W_i \in \mathscr{L}$, $1 \leqslant i \leqslant n$, because any $f \in W_i$ is learned by some $M_j \in \mathscr{T}'$ which then identifies the whole W_i.

Assume that for some S with $|S| \leqslant b$ we have $\bigcup_{i \in S} W_i \notin [1, b-1] \, \mathscr{L}$. This is possible only if $|S| = b$. For any $i \in S$ we denote $\mathscr{T}_i = \{M \in \mathscr{T}' \mid W_i \subseteq \mathscr{L}(M)\}$. We show that all $\mathscr{T}_i$, $i \in S$, are mutually disjoint. Indeed, if there is $M \in \mathscr{T}_{i_1} \cap \mathscr{T}_{i_2}$, then $W_{i_1} \cup W_{i_2} \in \mathscr{L}$, and we get $\bigcup_{i \in S} W_i \in [1, b-1] \, \mathscr{L}$, a contradiction. $\{\mathscr{T}_1, ..., \mathscr{T}_b\}$ becomes a b-partition of $\mathscr{T}'$ if we somehow distribute the remaining machines $\mathscr{T}' - (\mathscr{T}_1 \cup \cdots \cup \mathscr{T}_k)$ between the $\mathscr{T}_i$'s. Machines from all the families $\mathscr{T}_i$ are needed to identify

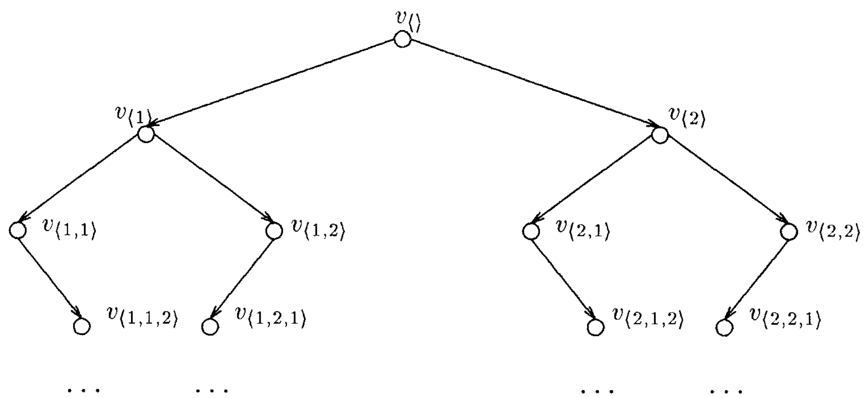

FIG. 1. Diversification tree.

$\bigcup_{i\in S} W_i$ along with as $W \supseteq \bigcup_{i\in S} W_i$. Hence, $\{\mathcal{T}_1, ..., \mathcal{T}_b\}$ is a (b, b)-diversification, a contradiction to the choice of $\mathcal{T}'$.

$(2)\Rightarrow(1)$ We use a diagonalization argument. Assume, that each set $W_S = \bigcup_{i\in S} W_i$ is identified by a team of $b-1$ machines: $M_S^1, ..., M_S^{b-1}$. Collecting such teams for any S, $|S| \leqslant b$, we get a finite collection of machines $\mathcal{T}'$. Assume $\{\mathcal{T}_1, ..., \mathcal{T}_b\}$ is a (b, b)-diversification of $\mathcal{T}'$. This means, that for any $\mathcal{T}_k$ there is a W_{i_k} such that the team $\mathcal{T}' - \mathcal{T}_k$ fails to identify some functions from W_{i_k}.

For each $k = 1, ..., b$ we find the corresponding i_k such that $\mathcal{T}' - \mathcal{T}_k$ does not learn W_{i_k}. Consider $S^* = \{i_1, ..., i_b\}$. Any $\mathcal{T}_k$, $1 \leqslant k \leqslant b$, should contain some $M^j_{S^*}$, for otherwise $\mathcal{T}' - \mathcal{T}_k$ would identify $W_{i_k} \subseteq \bigcup_{i\in S^*} W_i$. But there are only $b-1$ machines $M^j_{S^*}$, so not every $\mathcal{T}_k$ can get one, a contradiction. ∎

THEOREM 34. *Let $\mathcal{L}$ be some identification type, W some class of functions, and $b \geqslant 2$. The following statements are equivalent*:

1. *There is no $(2, b)$-diversification of all recursive machines for the class W.*

2. *W can be expressed as finite union $W = W_1 \cup \cdots \cup W_n$ such that*

$$\forall S \subseteq \{1, ..., n\} \left(|S| \leqslant b \Rightarrow \bigcup_{i\in S} W_i \in \mathcal{L}\right).$$

Proof. $(1)\Rightarrow(2)$ Assume that the collection of all IIMs $\mathcal{T}$ has no $(2, b)$-diversification for W. Using Lemma 32 we find a finite collection of machines $\mathcal{T}' = \{M_1, ..., M_m\}$ which has no $(2, b)$-diversification for W. Split the set W into equivalence classes $W_1, ..., W_n$; $f, g \in W$ are equivalent iff $f \in \mathcal{L}(M_j)$ implies $g \in \mathcal{L}(M_j)$ and vice versa for all $M_j \in \mathcal{T}$.

Assume that there is $S \subseteq \{1, ..., n\}$, $|S| = b$ such that $\bigcup_{i\in S} W_i \notin \mathcal{L}$. Then for any machine $M_j \in \mathcal{T}'$ there is a W_i, $i \in S$, such that M_j fails to identify W_i. Define a b-partition of $\mathcal{T}$ setting $\mathcal{T}_i = \{M \in \mathcal{T}' \mid W_i \nsubseteq \mathcal{L}(M)\}$ for all $i \in S$ and, if some M falls into several $\mathcal{T}_i$'s, delete it from all except one. We have obtained a $(2, b)$-diversification of $\mathcal{T}'$, a contradiction.

$(2)\Rightarrow(1)$ Assume that each set $W_s = \bigcup_{i\in S} W_i$, $|S| \leqslant b$, is learned by some machine M_S. Let $\mathcal{T}'$ be the collection of all such machines M_S. Assume by way of contradiction that there is a $(2, b)$-diversification of $\mathcal{T}'$, namely $\{\mathcal{T}_1, ..., \mathcal{T}_b\}$. This means that each team $\mathcal{T}_k$, $1 \leqslant k \leqslant b$, fails to learn the whole W. Let us say, that $\mathcal{T}_1$ fails to learn some W_{i_1}, $\mathcal{T}_2$ fails to learn W_{i_2}, etc. There is a machine M_{S^*} for the "diagonal" set $S^* = \{i_1, ..., i_b\}$. It does not belong to any member $\mathcal{T}_k$ of the $(2, b)$-diversification. Contradiction. ∎

It would be tempting to generalize both previous theorems to get the necessary and sufficient condition for the existence of an (a, b)-diversification for arbitrary a and b. It is known, however, that the condition

$$\forall S \subseteq \{1, ..., n\} \left(|S| \leqslant b \Rightarrow \bigcup_{i\in S} W_i \in [1, a-1]\ \mathcal{L}\right)$$

works only for $a = b$ (Theorem 33) and $a = 2$ (Theorem 34).

All the basic implications between (a, b)-diversifications are shown in Fig. 2. A directed path from (a_1, b_1) to (a_2, b_2) means that for any fixed class W and type $\mathcal{L}$ the existence of (a_1, b_1)-diversification implies that of (a_2, b_2)-diversification. For particular identification types there are nontrivial implications as well. This is the topic of the next subsections.

$$\begin{array}{ccccccccc}
(2,2) & \Rightarrow & (2,3) & \Rightarrow & (2,4) & \Rightarrow & (2,5) & \Rightarrow & \cdots \\
\Uparrow & & \Uparrow & & \Uparrow & & \Uparrow & & \\
(3,3) & \Rightarrow & (3,4) & \Rightarrow & (3,5) & \Rightarrow & \cdots & & \\
\Uparrow & & \Uparrow & & \Uparrow & & & & \\
(4,4) & \Rightarrow & (4,5) & \Rightarrow & \cdots & & & & \\
\Uparrow & & \Uparrow & & & & & & \\
\cdots & & \cdots & & & & & &
\end{array}$$

FIG. 2. The lattice of (a, b)-diversifications.

6.2. *Diversifications of EX Machines*

The comparisons between EX teams given by Theorems 7 and 8 allow a reformulation of the results of the previous subsection in a more convenient form.

LEMMA 35. *Let $W = W_1 \cup \cdots \cup W_n$ be a class of recursive functions expressed as a union of n classes. Let $k \geqslant 2$. Then the following statements are equivalent*:

1. $\forall S \subseteq \{1, ..., n\}$ $(|S| \leqslant k \Rightarrow \bigcup_{i \in S} W_i \in [1, k-1]\ EX)$,
2. $W \in [1, k-1]\ EX$.

Proof. Evidently $(2) \Rightarrow (1)$. Whenever $n \leqslant k$ we also get $(1) \Rightarrow (2)$. For $n = k+1, k+1, ...$, we prove $(1) \Rightarrow (2)$ inductively.

The base case is when $n = k+1$. Denote $U_j = \bigcup_{i \neq j} W_i$. Since $U_j \in [1, k-1]\ EX$, we have a team $\mathscr{T}_j$ of $k-1$ IIMs identifying U_j. Let $\mathscr{T} = \bigcup_{j=1}^{k+1} \mathscr{T}_j$; it is a collection containing $(k+1)(k-1)$ machines. A fixed function $f \in W$ belongs to all U_j excepting at most one. Therefore f is identified by at least k machines from the collection $\mathscr{T}$. So, $W \in [k, k^2-1]\ EX$, and by Theorem 8 we also have $W \in [1, k-1]\ EX$.

Suppose that the result holds for some $n \geqslant k$. We prove it for $n+1$. The inductive hypothesis yields $\bigcup_{i \in S} W_i \in [1, k-1]\ EX$ whenever $|S| \leqslant n$. Replace k by n and repeat the reasoning of the base case. ∎

THEOREM 36. *Let $k \geqslant 2$. A (k, k)-diversification for a class of recursive functions W exists iff $W \notin [1, k-1]\ EX$.*

Proof. Combine Theorem 33 and Lemma 35. ∎

COROLLARY 37. *Let $W = U_1 \cup U_2$ where $U_1, U_2 \in EX$ but $W \notin EX$. Then all recursive machines can be partitioned into two families such that W cannot be identified by a finite team taken from one family. Machines M_1 and M_2 identifying U_1 and U_2 respectively will be in different families.*

The following two results characterize the absence of uniqueness and universality for (k, k)-diversifications.

THEOREM 38. *For a fixed $W \notin [1, k-1]\ EX$ there are uncountably many (continuum) (k, k)-diversifications.*

Proof. Infinitely many machines which learn nothing can be added to or taken away from any family. ∎

$$
\begin{array}{ccccccccccc}
W \notin EX & \Leftrightarrow & (2,2) & \Leftrightarrow & (2,3) & \Leftrightarrow & (2,4) & \Leftrightarrow & (2,5) & \Leftrightarrow & \cdots \\
\Uparrow & & & & & & & & & & \\
W \notin [1,2]EX & \Leftrightarrow & (3,3) & \Leftrightarrow & (3,4) & \Leftrightarrow & (3,5) & \Leftrightarrow & \cdots & & \\
\Uparrow & & & & & & & & & & \\
W \notin [1,3]EX & \Leftrightarrow & (4,4) & \Leftrightarrow & (4,5) & \Leftrightarrow & \cdots & & & & \\
\Uparrow & & & & & & & & & & \\
\cdots & & & & & & & & & &
\end{array}
$$

FIG. 3. (a, b)-diversifications for the type EX.

THEOREM 39. *There exist two (disjoint) classes $W_1, W_2 \notin EX$ such that no 2-partition is a $(2, 2)$-diversification for both W_1 and W_2.*

Proof. By Theorem 10 there are two classes $U_1, U_2 \in EX$ such that $U_1 \cup U_2 \notin EX$. Let us make another pair of classes $V_1, V_2 \in EX$ and $V_1 \cup V_2 \notin EX$ such that functions $f \in U_1 \cup U_2$ are discernable from functions $g \in V_1 \cup V_2$. We can require, for example, that $f(0) = 0$ and $g(0) = 1$.

There exist machines M_1, M_2, M_3, M_4 identifying the unions $U_1 \cup V_1$, $U_1 \cup V_2$, $U_2 \cup V_1$, $U_2 \cup V_2$ respectively. We can easily check that for any partition of machines M_1, M_2, M_3, M_4 into two subfamilies, one of these subfamilies will identify either $W_1 = U_1 \cup U_2$ or $W_2 = V_1 \cup V_2$. ∎

In contrast with the FIN-learning example mentioned at the very beginning of this section, diversification results for EX learning are not caused by inability of separate families in a (k, k)-diversification to learn certain functions from the target class W. On the contrary, each family in the valid (k, k)-diversification provides machines for learning any total recursive function in the following sense:

LEMMA 40. *Let $W \in [1, n]\ EX$ and $\{\mathscr{T}_1, \mathscr{T}_2 ..., \mathscr{T}_k\}$—a (k, k)-diversification for W. Then, for any r.e. closed class $V \subset \mathscr{F}$, all the members $\mathscr{T}_i$ of the (k, k)-diversification contain machines learning V.*

Proof. Let $M_1, M_2, ..., M_n$ be a team EX-learning W and let $M'_1, M'_2, ..., M'_n$ be machines, produced by Theorem 12, such that $EX(M'_i) = EX(M_i) \cup V$, $i = 1, 2, ..., n$. Suppose that some $\mathscr{T}_j$ does not contain any M'_i. Then we can pick team

$$M'_1, M'_2, ..., M'_n$$

EX-learning W and disjoint from $\mathscr{T}_j$, which contradicts the assumption that $\{\mathscr{T}_1, \mathscr{T}_2, ..., \mathscr{T}_k\}$ is a (k, k)-diversification. Therefore any $\mathscr{T}_j$ contains some M'_i which EX-learns V. ∎

Any (k, k))diversification for W, obviously is a (k, k)-diversification for all supersets $W' \supset W$. The following result shows that it is a (k, k)-diversification for some subsets of W as well.

THEOREM 41. *If there is a (k, k)-diversification for a class W then the same (k, k)-diversification is also for $W' = W - V$ whenever V is r.e. and closed.*

Proof. If W is not learnable by a finite team, the assertion is trivial —any k-partition of all machines is a (k, k)-diversification for both W and $W - V$. Assume that $W \in [1, n]\ EX$ and some (k, k)-diversification for W is not (k, k)-diversification for $W - V$. Pick a team $\mathscr{T}'$ learning $W - V$ which does not intersect all members of the (k, k)-diversification. According to the previous lemma we can add to $\mathscr{T}'$ some machine M learning V, where M can be picked from some $\mathcal{T}_j$ already intersecting $\mathcal{T}'$. Team $\mathcal{T}' \cup \{M\}$ EX-learns W and still does not intersect all members of the (k, k)-diversification, a contradiction. ∎

6.3. *Diversifications of EX_n Machines*

Types EX_n have weaker voting properties than EX (cf. Theorem 9 with Theorem 8). Therefore we can guarantee weaker diversifications for EX_n machines (cf. Theorem 43 with Theorem 36).

LEMMA 42. *Let $W = W_1 \cup \cdots \cup W_m$ be a class of recursive functions expressed as a union of m classes. The following statements are equivalent*:

1. $\forall S \subseteq \{1, ..., m\}(|S| \leqslant 2^{n+2} - 1 \Rightarrow \bigcup_{i \in S} W_i \in EX_n)$,
2. $W \in EX_n$.

Proof. Parallel to that of Lemma 35. ∎

THEOREM 43. *A* $(2, 2^{n+2} - 1)$*-diversification of all EX_n machines for a class of recursive functions W exists iff* $W \notin EX_n$.

Proof. Combine Theorem 34 and Lemma 42. ∎

Since $FIN = EX_0$, any $W \notin FIN$ has a $(2, 3)$-diversification. In contrast with the type EX there are classes $W \notin FIN$ for which here is no $(2, 2)$-diversification. In fact, any $W \in ([2, 3]\ FIN - FIN)$ will do.

THEOREM 44. *If $W \in [k, m]\ FIN$ where $k/m > 1/2$, then there is no* $(2, 2)$*-diversification of all FIN machines for W.*

Proof. Let $\mathcal{T} = \{M_1, ..., M_m\}$ be a team identifying W in the sense $[k, m]\ FIN$. For every $f \in W$ we determine those M_i which identify f. We split W into a union of disjoint subclasses, where two functions are in the same W_j whenever they are identified by the same subset of $\mathcal{T}$. The union of any two subclasses $W_{j_1} \cup W_{j_2}$ is identifiable by a single machine from the team. Indeed, both W_{j_1} and W_{j_2} are identified by more than a half of all members of the team, and, consequently, the two sets of machines must intersect. Apply Theorem 33 to complete the proof. ∎

The existence of a $(2, 2)$-diversification for $W \notin FIN$ depends, among other things, upon whether or not W contains its accumulation points.

DEFINITION 45. Function $f \in \mathcal{F}$ is an *accumulation point* of a $W \subseteq \mathcal{F}$ if any neighborhood of f intersects $W - \{f\}$.

In other words, f is an accumulation point of W iff by seeing finitely many values of f we cannot distinguish it from some other function $f_1 \in W$. The f itself may or may not belong to W. Tradition compels us to use the word "point" when referring to the functions from topological spaces.

DEFINITION 46. A function $f \in W$ is called an *isolated point* in W if it is not an accumulation point of W.

THEOREM 47. *If W contains an accumulation point of itself, then there exists a* $(2, 2)$*-diversification of all FIN machines for W.*

Proof. Let $f \in W$ be an accumulation point of W. The first family of a $(2, 2)$-diversification $\mathcal{T}_1$ contains all machines which identify f. $\mathcal{T}_2$ contains all the other machines. Any finite team $\mathcal{T}'$ learning W must intersect $\mathcal{T}_1$. Now suppose from the contrary that $\mathcal{T}'$ does not intersect $\mathcal{T}_2$. Each machine from $\mathcal{T}'$ outputs an index of f after receiving a finite string σ of values of f. Intersection of all the corresponding neighborhoods U_f^σ of f is again a neighborhood of f. As f is an accumulation point, there is $f_1 \in W$ in this neighborhood such that $f_1 \neq f$. The function f_1 is not identified by $\mathcal{T}'$, a contradiction. ∎

COROLLARY 48. *If $W \in [k, m]\ FIN$ where $k/m > 1/2$, then all functions in W are isolated.*

Proof. Combine Theorems 44 and 47. ∎

7. CONCLUSIONS

Felix Klein's *Erlagen Program* [Wey52] introduced a highly successful research paradigm to the mathematics community. The basic idea is to look at the inverse of a problem and study transformations. In essence, all work to date in inductive inference has taken a machine based approach to ask, "Given an IIM, what functions can it learn?" We propose to follow the Erlagen Program and ask, "Given a function, which IIMs can learn it?" Some dualities with conventional notions were discovered. The dualities however are not complete as made clear by the comment at the end of Section 5.

Finite sets have a well known property—it is possible to add them to any class without affecting its EX-learnability. Theorem 12 shows that infinite sets that are closed and recursively enumerable have this property as well. This is reminiscent of descriptions of compact sets in real analysis as closed and bounded subsets of Euclidean space. If the description of set V in Theorem 12 as "closed and recursively enumerable" will eventually be replaced by some less restricting property, then results equivalent to Theorem 12, e.g., Lemma 40 and Theorem 41 and maybe also Theorem 13, will be modified accordingly.

Theorem 8 states, for instance, that $[2, 3]\ EX = EX$. Actually we can construct an operator Φ such that whenever $M = \Phi(M_1, M_2, M_3)$, machine M EX-learns the same class of functions which is $[2, 3]\ EX$-learned by the team $\{M_1, M_2, M_3\}$. Let $(\mathcal{T}_1, \mathcal{T}_2)$ be a $(2, 2)$-diversification of all IIMs for some class $W \in ([1, 2]\ EX - EX)$. Suppose that M_1, M_2, M_3 learn a substantial quantity of functions from $W \notin EX$, i.e., $W - EX(M_i) \in EX$, $i = 1, 2, 3$. Then $M = \Phi(M_1, M_2, M_3)$ also has the property $W - EX(M) \in EX$ and belongs to the same side of partitioning $(\mathcal{T}_1, \mathcal{T}_2)$ to which majority of M_1, M_2, M_3 belongs. Therefore, the "voting operator" Φ actually performs voting not only on the level of individual machines, but also on the level of partitions relative to a given $W \notin EX$.

Received October 5, 1995; final manuscript received June 24, 1996

REFERENCES

[AFK⁺92] Apsītis, K., Freivalds, R., Kriķis, M., Simanovskis, R., and Smotrovs, J., Unions of identifiable classes of total recursive functions, *in* "Analogical and Inductive Inference" (K. Jantke, Ed.), pp. 99–107, Lecture Notes in Artificial Intelligence, No. 642, Springer-Verlag, Berlin/New York, 1992.

[AFS94] Apsītis, K., Freivalds, R., and Smith, C., Choosing a learning team: A topological approach, *in* "Proceedings of the 26th Symposium on the Theory of Computing," pp. 283–289, Assoc. Comput. Mach., New York, 1994.

[AGS89] Angluin, D., Gasarch, W. I., and Smith, C. H., Training sequences, *Theoret. Comput. Sci.* **66** (1989), 255–272.

[AS83] Angluin, D., and Smith, C. H., Inductive inference: Theory and methods, *Comput. Surv.* **15** (1983), 237–269.

[Bar74] Barzdins, J., Two theorems on the limiting synthesis of functions, *in* "Theory of Algorithms and Programs" (J. Barzdins, Ed.), Vol. 1, pp. 82–88, Latvian State University, Riga, 1974.

[BB75] Blum, L., and Blum, M., Toward a mathematical theory of inductive inference, *Inform. and Control.* **28** (1975), 125–155.

[CS83] Case, J., and Smith, C., Comparison of identification criteria for machine inductive inference, *Theoret. Comput. Sci.* **25**(2) (1983), 193–220.

[DKV92] Daley, R., Kalyanasundaram, B., and Velauthapillai, M., Breaking the probability 1/2 barrier in FIN-type learning, *in* "Proceedings of the Fifth Annual Workshop on Computational Learning Theory" (L. Valiant and M. Warmuth, Eds.), pp. 203–217, Assoc. Comput. Mach., New York, 1992.

[DKV93] Daley, R., Kalyanasundaram, B., and Velauthapillai, M., Capabilities of fallible finite learning, *in* "The 1993 Workshop on Computational Learning Theory," The Association for Computing, 1993.

[DPVW91] Daley, R., Pitt, L., Velauthapillai, M., and Will, T., Relations between probabilistic and team one-shot learners, *in* "Proceedings of the 1991 Workshop on Computational Learning Theory," Palo Alto, CA, 1991, (M. Wormuth and L. Valiant, Eds.), pp. 228–239, Morgan Kaufmann, San Mateo, CA, 1991.

[Eng89] Engelking, R., "General Topology," Heldermann Verlag, Berlin, 1989.

[FS93] Freivalds, R., and Smith, C., On the duality between mechanistic learners and what it is they learn, *in* "Lecture Notes in Artificial Intelligence," Vol. 744, pp. 137–149, Springer-Verlag, Berlin/New York, 1993.

[FSV89] Freivalds, R., Smith, C., and Velauthapillai, M., Trade-offs amongst parameters effecting the inductive inferribility of classes of recursive functions, *Inform. and Comput.* **82** (1989), 323–349.

[Gol67] Gold, E. M., Language identification in the limit, *Inform. and Control.* **10** (1967), 447–474.

[KF92] Kriķis, M., and Freivalds, R., Inductive inference of total recursive functions by probabilistic and deterministic strategies, *in* "Technical Report YALEU/DCS/TR-936," November 1992.

[Kur58] Kuratowski, C., "Topologie," 4th ed., Monographie Matematyczne, Vols. 20–21, Panstwowe Wydawnictwo Naukowe, Warsaw, 1958.

[KZ91] Kinber, E., and Zeugmann, T., One-sided error probabilistic inductive inference and reliable frequency identification, *Inform. and Comput.* **92** (1991), 253–284.

[MY78] Machtey, M., and Young, P., "An Introduction to the General Theory of Algorithms," North-Holland, New York, 1978.

[OSW86] Osherson, D., Stob, M., and Weinstein, S., "Systems That Learn," MIT Press, Cambridge, MA, 1986.

[Pit89] Pitt, L., Probabilistic inductive inference, *J. Assoc. Comput. Mach.* **36**(2) (1989), 383–433.

[PS88] Pitt, L., and Smith, C., Probability and plurality for aggregations of learning machines, *Inform. and Comput.* **77** (1988), 77–92.

[Rog67] Rogers, H., Jr., "Theory of Recursive Functions and Effective Computability," McGraw–Hill, New York, 1967.

[Sch86] Schafer, G., Some results in the theory of effective program synthesis: Learning by defective information, *in* "Lecture Notes in Computer Science," Vol. 215, Springer-Verlag, Berlin/New York, 1986.

[Smi82] Smith, C. H., The power of pluralism for automatic program synthesis, *J. Assoc. Comput. Mach.* **29**(4) (1982), 1144–1165.

[Smi94a] Smith, C., "A Recursive Introduction to the Theory of Computation," Springer-Verlag, Berlin/New York, 1994.

[Smi94b] Smith, C., Three decades of team learning, *in* "Proceedings of AII/ALT'94," Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin/New York, 1994.

[Wey52] Weyl, H., "Symmetry," Princeton Univ. Press, Princeton, NJ, 1952.

[Wie74] Wiehagen, R., Inductive inference of recursive functions, *in* "Lecture Notes in Computer Science," Vol. 32, pp. 462–464, Springer-Verlag, Berlin/New York, 1974.